Mediators in Infrastructure Survivability Enhancement

Kevin J. Sullivan
University of Virginia
Computer Science Department
Thornton Hall
+1 804 982 2206
sullivan@Virginia.EDU

Steve Geist
University of Virginia
Computer Science Department
Thornton Hall
+1 804 982 2200
smg9c@Virginia.EDU

Paul Shaw
University of Virginia
Computer Science Department
Thornton Hall
+1 804 982 2200

1. ABSTRACT

A key research priority for the next decade is the protection of critical, software-intensive infrastructures—e.g., electric power, banking, telecommunications, and transportation. The problem is complicated by the need to enhance existing systems. We describe one approach to *survivability enhancement*. In 1997 the Internet failed when corrupt data was disseminated at the top level of the Domain Name System. We replicated this failure and developed a solution based on transparent insertion of mediators to enforce survivability policies. Our approach promises to ease survivability enhancement in two ways: transparent insertion eases system architectural evolution; and modularization of survivability policy implementations eases the evolution of both survivability policies and the systems into which our mediators are inserted.

1.1 Keywords

mediator, infrastructure, survivability, internet, DNS

2. INTRODUCTION

The survivability of software-intensive infrastructures—e.g., electric power, transportation, banking and finance and telecommunications—has emerged as a major concern of government and industry and as an important research topic [3][4][5][6]. By survivability, we mean, informally, assured continuity of essential infrastructure services under defined adverse conditions: natural, accidental or hostile. Society depends on critical infrastructures for heat, light, food, emergency services, commerce, etc. The use of software-intensive subsystems within infrastructures creates enormous value, but such software systems are also complex and fragile. They support multiple functions; are highly distributed; sometimes reactive; in some cases have real-time requirements; and they embed a great deal of domain-specific knowledge. The deep and rapidly growing reliance of critical infrastructures on such information systems thus puts society at risk.

For example, the transition to the Internet as a medium for commerce provides for enormous efficiency in economic transactions; yet, the Internet is fragile and thus prone to catastrophic failure. Failures that have already occurred make it clear that the risks are not merely speculative [10].

The problem of the survivability of information-intensive infrastructures is compounded by their reliance on complex, aged software—i.e., legacy systems—and on extensive use of commercial-off-the-shelf (COTS) components. Unfortunately, redesigning such systems from scratch for survivability is untenable. Recent experiences with the United States Internal Revenue Service, air traffic control, and other such critical systems show that we do not necessarily have the intellectual or monetary capital to succeed in such demanding tasks [3].

On the other hand, evolving complex existing systems presents its own challenges. Yet, given that we do now and will continue to depend on existing systems, *survivability enhancement* emerges as an important topic. Of particular interest are enhancement techniques operating at the architecture level, because it is at this level that key system-wide properties are enabled, if not assured.

We report on an experimental systems project undertaken within a broader program on survivability architecture [1][2][9]. The infrastructure is the Internet. We studied the embedded information system called the Domain Name Service (DNS). The failure of DNS in 1997 brought down a large part of the Internet [10]. We describe the survivability enhancement of DNS by the transparent architectural insertion of a *survivability mediator* into DNS. The mediator approach [7][8] was developed to ease the design and evolution of complex integrated systems.

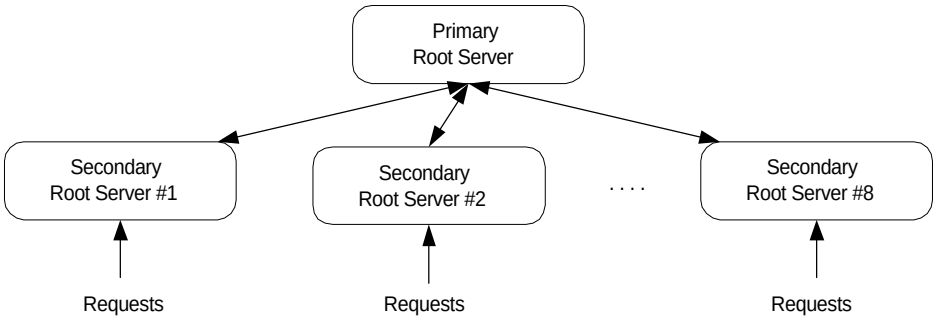

Figure 1. Basic architecture of the root level of the internet domain name service

3. BACKGROUND

DNS is a distributed application that enables Internet addressing based on mnemonic Internet host names, such as *www.virginia.edu.* DNS resolves such names to numerical addresses, such as *128.143.136.15*, that network routers can use directly in the routing of Internet Protocol (IP) packets.

DNS is a hierarchically structured, distributed system that manages a database of name-to-address mappings. The *root level* of the hierarchy, illustrated in Figure 1, provides translations for partial names such as *.com* and *.net*. When a user requires a name translation, the local DNS database is queried. If there is no local translation, the query is sent up the hierarchy, eventually reaching a secondary root server. If no translation is found there, the user receives an error message saying that the name could not be resolved.

The software running on both primary and secondary server nodes is called Berkley Internet Name Domain (BIND). BIND is implemented in about 30,000 lines of C code. By any measure BIND is a simple infrastructure information system. Yet is has attractions as a subject for research. First, it is a real system. Second, unlike most infrastructure information systems, it is amenable to study in a research setting. Versions of the code are widely available and written in C. Finally, it is the software for a real critical infrastructure, the failure of which lead to a massive outage. The DNS size is not representative, but DNS is nevertheless useful as a specimen for experimental systems research.

The set of existing Internet names changes continually. The primary and secondary servers, running BIND, engage in a protocol that results in top-level database updates being propagated from the primary to secondary servers (arrows in Figure 1). The 1997 failure occurred when, owing to an operator ignoring an alarm, the primary server at Network Solutions, Inc. (NSI) passed an empty database to eight secondary root servers. This failure disabled the root level translation of names ending in *.com* and *.net*. Because these servers serve most of the Internet, the data corruption effectively erased the .com and .net Internet domains.

The error occurred at 2:30 AM EST and so had a greater impact in Europe than in the United States. However, over one million companies have domains within .com and .net. The primary server database was corrected by 6:30 AM EST, but the secondary servers had to manually reload the "zone data." The effects of the problem lasted into the afternoon [4]. The damage was not great owing to the still limited dependence of the economy on the Internet and to the timing of the outage. However, that such a massive failure could result from a simple operator error evinces the fragility of information-intensive infrastructures.

Although the scale of DNS is small, it failure modes are representative. The failures that we pinpoint are operator error and dissemination of corrupted information within an infrastructure. The DNS failure also shows that traditional techniques for improving reliability are inadequate to assure infrastructure survivability. First, data corruption occurred when a human operator at NSI caused a mangled database to be sent to secondary servers despite an alarm indicating that there was a problem. An accidental error or malicious act by one person should not be allowed to lead to massive infrastructure outages. Second, the traditional hardware approach to availability through replication—in this case in the form of secondary servers—was of no potential use. The *data* were bad, not the systems for manipulating them.

4. APPROACH

The approach to survivability enhancement that we have explored is based the insertion of *survivability mediators* into infrastructure information systems at the architectural level. A mediator is a software module that is interposed transparently between other modules of a system, and that serves two purposes, one functional, one evolutionary. First, a mediator enforces a specified behavioral relationship among the mediated modules. Second, the mediator eases evolution in two ways. Transparent insertion eases architectural evolution; and the information hiding encapsulation of the implementation of the specified behavioral relationship permits the relationship and the rest of the system to evolve largely independently of each other.

A *survivability mediator* is a mediator that enforces a policy that satisfies a survivability requirement. The

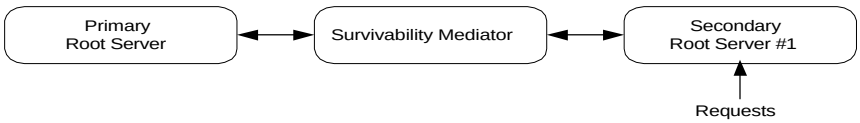

Figure 2. Simplified architecture of DNS hardened with a survivability mediator

survivability requirement that we chose to explore is one that could have averted the Internet failure that actually occurred: *DNS must continue to provide the domain name service to users, albeit perhaps with degraded accuracy, even in the face of corrupted data being sent from the primary root server.*

Figure 2 illustrates the mediator-based architectural change that we made to effect this survivability enhancement. We introduced a mediator between the primary and each of the secondary root servers. The figure shows the change for just one secondary server. Because DNS servers running BIND communicate using TCP/IP, and because the interconnection of BIND nodes is programmed in text-based configuration files, inserting the mediator required only that these files be changed to cause BIND nodes to communicate through the mediator rather than directly.

The mediator waits for connections from the primary or the secondary server, buffers the message being sent, then implements policy actions depending on the content seen. For instance, in an early prototype, if the database file that is sent from the primary to the secondary is found to be empty, the mediator discards the message.

Unfortunately, this policy turned out not to meet the desired survivability requirement. Merely quashing messages is not enough, because database entries on secondary servers time out. In the absence of periodic updates from the primary, the name service provided by the secondary servers evaporates quickly. The requirement is thus that secondary servers see a steady stream of valid updates.

Our next policy was to have the mediator maintain a copy of the last valid database sent from the primary to the secondary, and to re-send the copy if a corrupted database were detected. This policy was also inadequate. A secondary server requests an update from the primary when its sees that its cached databases is older than the one available on the primary. When it receives a new one from the primary, it checks its contents to ensure that it bears a time-stamp matching the one on which the request was based. A mere copy the old database is not valid.

The policy that finally worked was to send the old database *with appropriate edits* to ensure that the secondary accepts it as valid. Our mediator thus forwards clever counterfeits.

In addition to recognizing empty databases as corrupt, we explored policies based on unexpected changes in size. The point is not that complex policies are needed. That an alarm was sounded at NSI but ignored in the actual incident shows that error detection was not the problem. The problem was in *enforcing* a critical integrity constraint. Anticipating data corruption as an important failure mode, assessing the risk of occurrence, and taking appropriate actions at the architectural design level shows promise as a strategy for infrastructure survivability enhancement.

5. CONCLUSION

Our mediator approach eased the evolutionary survivability enhancement of a critical infrastructure system by allowing us to impose a new integrity constraint without changes to the existing source code. Enhancement will not be so easy for all systems; but transparent insertion of mediators at natural architectural boundaries is a general strategy independent of the type of boundary: CORBA procedure invocation, Internet event notification, file system call, etc.

The mediator approach has the added benefit of localizing the survivability policy implementation, making it possible to change both it and the surrounding system independently. We exploited this dimension of evolvability in exploring a range of policy specifications and implementations. In general, policy evolution promises to be important for future infrastructure information systems—e.g., to keep pace with increasing capabilities of hostile adversaries.

Critical systems with policy enforcement mechanisms often have overrides. A nuclear-powered submarine has a *battle short* to enable necessary but dangerous power generation in combat. Such features in infrastructures (e.g., ability of managers to approve transmission of corrupted data) should perhaps be explicit in survivability requirements. Adding *except in case of management override* to our policy for DNS might make sense, for example. This extension could be made with local changes to our survivability mediator.

One of the authors (Shaw) developed the first survivability mediator for DNS as part of a senior undergraduate thesis project. A major part of the effort was in his understanding BIND—30K lines of arcane TCP/IP networking code. In particular, it was necessary to find

the hooks to manipulate the BIND communication streams. If Shaw been an expert, he would have found the hooks quickly. On the other hand, his experience gives an indication of the difficulties that probably await those undertaking survivability enhancement of large and poorly documented infrastructures. In the end we only had to understand in depth those parts of BIND that are involved in zone (database) updates. Our mediator code is much less complex than BIND: about 3,000 lines.

"Reifying connectors" is not a new idea. It is the heart of the mediator concept, for example [7][8]. In this paper, we presented a case study on adapting this technique to the survivability enhancement of critical infrastructures. In an experimental systems study, we applied the technique to mitigate the failure mode exhibited in the 1997 catastrophic failure of one critical infrastructure—the Internet.

6. ACKNOWLEDGMENTS

Work was sponsored by the Defense Advanced Research Projects Agency and Rome Laboratory, Air Force Materiel Command, USAF, under agreement number F30602-96-1-0314. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purpose notwithstanding any copyright annotation thereon. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either express or implied, of the Defense Advanced Research Projects Agency, Rome Laboratory or the U.S. Government. Support was also provided by the National Science Foundation under grants CCR-9502029, CCR-9506779 and CCR-9804078. We acknowledge discussions with John Knight in which he emphasized the dissemination of corrupted data as an important failure mode for critical infrastructure systems. We thank Jorg Liebeherr for commenting on an earlier version of this paper.

7. REFERENCES

[1] J.C. Knight, R. W. Lubinsky, J. McHugh, and K. J. Sullivan, Architectural Approaches to Information Survivability, Technical Report CS-97-25, Department of Computer Science, University of Virginia, Charlottesville, VA 22903 (September 1997).

[2] J. C. Knight, M. C. Elder, J. Flinn, and P. Marx, Summaries of Three Critical Infrastructure Applications, Technical Report CS-97-27, Department of Computer Science, University of Virginia, Charlottesville, VA 22903 (December 1997).

[3] National Coordination Office for Computing, Information and Communications, *President's Information Technology Advisory Committee: Interim Report to the President*, August, 1998.

[4] Office of the Press Secretary, The White House, "Protecting America's Critical Infrastructures: PDD 63," May 22, 1998.

[5] Office of the Undersecretary of Defense for Acquisition & Technology, *Report of the Defense Science Board Task Force on Information Warfare-Defense (IW-D)*, November 1996.

[6] President's Commission on Critical Infrastructure Protection, *Critical Foundations: Protecting America's Infrastructures*, United States Government Printing Office (GPO) number 040-000-00699-1.

[7] Sullivan, K.J. and D. Notkin, "Reconciling Integration and Ease of Evolution," *ACM Transactions on Software Engineering and Methodology*, vol. 1, no. 3, July 1997.

[8] Sullivan, K.J. *Mediators: Easing the design and Evolution of Integrated Systems.* Ph.D. Dissertation, University of Washington Department of Computer Science and Engineering, Technical Report UW-CSE-94-08-01, 1994.

[9] Sullivan, K.J., J.C. Knight, Xing Du and S. Geist, "Information Survivability Control Systems," University of Virginia Department of Computer Science Technical Report CS-98-30, 1998 (submitted for publication).

[10] Wayner, P. *Human Error Cripples the Internet*. The New York Times, July 17, 1997.